A DISTRIBUTED ADAPTATION MODEL FOR AMBIENT ASSISTIVE LIVING APPLICATIONS

M. T. Segarra
IT / TELECOM Bretagne, Technopôle Brest-Iroise, Brest Cedex 3, France

F. André
IRISA / INRIA Rennes-Bretagne Atlantique, Campus de Beaulieu, Rennes, France

Keywords: Distributed adaptation model, Adaptive software, Ambient assistive living application.

Abstract: Ambient assistive living applications are generally intended to allow older people to easily access medical and emergency services, and maintain social and relatives contacts. In order to improve performance, elderly related data may be distributed among the medical staff and/or the relatives. However, access to such data may require different properties such as confidentiality or urgency depending on the actor accessing data, the computing environment used to access them, and their nature. Therefore, ambient assistive living applications should be designed as adaptive software and include the necessary mechanisms to dynamically modify their behavior. In this paper, we propose a model for dynamic adaptation that clearly separates adaptation from business logic and that can be customized by applications designers in order to satisfy adaptation needs. The model is based on a set of mandatory functionalities that manage basic adaptation operations and an optional functionality that can be used to distribute adaptation mechanisms.

1 INTRODUCTION

Ambient assistive living applications are generally intended to allow older people to stay at home in a familiar environment while connected to the family and medical staff. Such applications include several types of services such as health care services, communication between the elderly and their family, information-oriented services, etc. all of them allowing the concerned people to access elderly-related data. Access to such data may require different properties depending on the nature of the data or even the actor accessing them. The computing environment (fixed or wireless network, portable or fixed workstation...) may also influence access conditions. We consider such applications as a set of logical environments: the medical environment, the family members environment, etc. Each environment may be distributed and has different adaptation needs (emergencies, security, etc). Therefore, ambient assistive living applications should be designed as adaptive software and include the necessary mechanisms to dynamically modify their behavior.

Dynamic adaptation has recently become an important issue when designing and developing applications. Indeed, applications may execute on different heterogeneous environments which available resources vary over time. Research efforts are mainly focused on the proposal of basic tools for dynamic adaptation. These tools allow applications to specify adaptation policies (interesting changes on resources and required adaptation actions) and according to them include all the necessary machinery for deciding, planning and executing adaptation actions. However, the proposed tools do not take into account distribution neither of components nor of adaptation mechanisms themselves which limits their scope.

Our current research work focus on extending adaptation mechanisms so that adaptation designers are able to choose a distributed architecture for their system. In this sense, we aim at proposing a generic adaptation model that offers a wide set of functionalities that can be customized by adaptation designers.

The paper is organized as follows. Next section introduces the application we have considered in order to propose our adaptation model. Section 3 presents the terms we use concerning adaptation and introduces the basic functionalities that should be imple-

T. Segarra M. and André F. (2009).
A DISTRIBUTED ADAPTATION MODEL FOR AMBIENT ASSISTIVE LIVING APPLICATIONS.
In *Proceedings of the International Conference on Health Informatics*, pages 363-368
DOI: 10.5220/0001774403630368

mented to make a software adaptive. Section 4 extends the basic functionalities in order to add flexibility when building an adaptive distributed architecture. Finally, sections 5 and 6 position our work in the research results concerning adaptation and discuss future work, respectively.

2 TARGETED APPLICATION

The growth in population ageing in most industrialized nations is posing several problems at different levels of our society (political, social, and familiar/personal levels). Statistics data provide a clear picture of the problem dimensions. According to recent results of a survey funded by the European Commission (Börsch-Supan and Jürges, 2005), in 2050 the share of the above 60 age group will be around 37 in Europe and even more in Japan, and slightly lower in North America (27%).

Improving the quality of life of such a population depends on the efficiency, comfort and cosiness of the place an individual calls "home". Indeed, the prevalence of disability increases with age at a significant rate (Commission, 2007). As a result of the shifting age profile, the size of the impaired population will increase significantly in future years. For the elderly, home is a place of memories where they spend a lot of their time. Their demands on their home environment will increase and change with growing age, especially when their state of health starts to worsen. Yet the ability to perform the activities of daily life with no or little help from other, is essential to older people's well-being and self-esteem. And active participation in society, through social contacts and activities, daily economic activities such as shopping, and democratic decision-making are key to well-being.

Therefore, an assistive living application has to offer a set of services that allow older people to live longer at the place they like most, while ensuring their autonomy and high quality of life. We consider this application as composed of a set of services that can be grouped into three categories:

- health-oriented services that allow the elderly to access to medical services, and facilitate collaboration among medical staff,
- communication-oriented services that allow the elderly to maintain social contacts,
- information-oriented services that allow the elderly to access information to which they are interested in.

Services on different categories do not require the same properties of the execution environment such as security, confidentiality or urgency. For example, an alert may require a general practitioner to urgently establish a high-quality video conference contact with the elderly in a secure manner while a less-secured, lower-quality link may be establish between the family and the elderly. Therefore, changes on the execution environment (e.g. bandwidth), should lead the application to perform adaptation actions so that the required properties are maintained. Moreover, urgency and preferences must be taken into account when deciding adaptation actions. For example, an urgent situation may need the utilization of all available bandwidth resources leading all other services to be stopped.

On the other hand, disease(s) of an elderly person evolve over time. Time scale of this evolution varies from several minutes (e.g. an older person reading local news) to several years (e.g. Alzeihmer's disease). An ambient assistive living application has to adapt to the elderly capacities and dynamically change available services according to them. For example, a vocal synthesis service may be used when the older is exhausted so that she is able to continue be informed about local news.

Therefore, adaptation of an ambient assistive living application should cope with resources variability in order to ensure quality of service, and elderly capabilities that may vary over time. Traditionally, adaptation has been included in an ad-hoc manner leading to specific development efforts and limiting the interest of the provided services. Our approach is to propose a generic model for adaptation mechanisms that can be customized to particular adaptation needs. It is structured as a set of functionalities that allows for monitoring execution context, deciding about adaptation, planning adaptation actions and executing them. We have been using our model for adapting centralized applications and we are currently considering the ambient assistive living application and its distributed services in order to analyze the impact of distribution on our model.

3 ADAPTIVE SOFTWARE ARCHITECTURE

3.1 Vocabulary

In order for a software to be adapted, one may decide where adaptation mechanisms are associated to: the whole application, processes, services, components, or data. We consider each of them, that we call *entity*, as composed of a set of, potentially distributed,

components that may manage data. Therefore, a site[1] may execute a whole entity or part of it. For example, our ambient assistive living application may offer a health care service that 1) manages patients medical record and 2) includes a prescription sub-service. The whole health care service, each patient record as well as the prescription sub-service are called entities. Therefore, an adaptation designer may choose to associate adaptation mechanisms to the whole service, to each patient record, to the prescription sub-service or to all three, leading to a hierarchical organization of adaptation mechanisms.

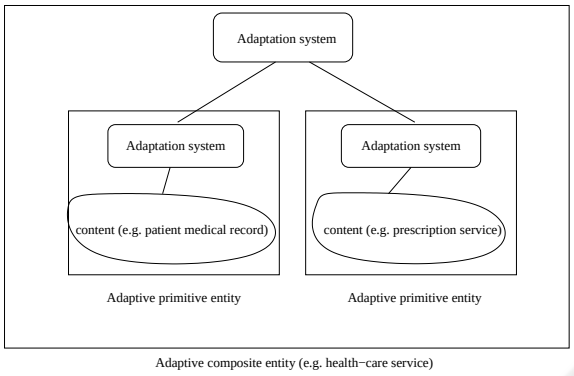

Figure 1: Adaptive composite entities, adaptive primitive entities.

An entity may include adaptation mechanisms while an adaptive (or self-adaptive entity) should include them. As depicted in Figure 1, adaptive entities may be primitive or composite. Adaptation mechanisms of primitive adaptive entities implement adaptation functionalities for the entity. Adaptation mechanisms of composite adaptive entities coordinate the adaptation of the included adaptive entities.

We call *adaptation model* the set of generic functionalities and their relation that ensure adaptation of an adaptive entity. Some of them will be mandatory but others will not. Specializing a functionality leads to a particular adaptation service. Finally, the set of adaptation services that implement adaptation of a particular entity is called an adaptation system.

3.2 Mandatory Adaptation Model: The DYNACO Framework

Previous work on adaptation have lead to DYNACO (Buisson, 2006)[2], a framework that aims at helping developers designing and implementing adaptation into their own applications. The framework is expressed as an assembly of abstract FRACTAL (Bruneton et al., 2006) components. Developers are expected to specialize them in order to build a framework instance specific to each application.

As Figure 2 shows, DYNACO decomposes adaptation mechanisms into four sub-functionalities:

- **observe:** DYNACO has to monitor execution conditions; the detection of any relevant change triggers adaptation;
- **decide:** given that something has changed in the execution environment, the framework has to find out the best strategy the entity should use to ensure that QoS properties of the service are observed;
- **plan:** once a strategy has been decided, it has to be implemented into a plan, which orders the actions to change the behavior that will make the adaptive entity achieve the strategy;
- **execute:** at last, planned actions have to be scheduled and executed, taking care of the synchronization with the execution of the entity code.

Figure 2: Overview of the DYNACO component assembly.

In order to foster effective code reuse, DYNACO encourages that those sub-functionalities are implemented thanks to existing generic engines, which are specialized specifically to the adaptive entity. DYNACO does not restrict to any particular engine, and we have already experimented several of them. For instance, the *decide* component has been implemented as a delegation to a Java virtual machine, a wrapper of the Jess (jes,) rule-based expert system, and a genetic algorithm.

Similarly, the *plan* component can be implemented either as a straightforward mapping from strategies to plans or as a logic-based planning algorithm such as STRIPS (Fikes and Nilsson, 1971).

The *execute* component depends on the application programming model. For example, we have proposed AFPAC (Buisson et al., 2006) as an implementation for SPMD applications but others can be used as well.

Finally, the *observer* component mainly depends on the execution environment, rather than on the application or the adaptation. Consequently, we consider that it is provided by the execution environment.

As the framework does not depend on the adaptation actions, and as the dependency to the program-

[1] A set of computers managed by the same administration.

[2] The DYNACO framework is available at the following web site: `http://dynaco.gforge.inria.fr/`

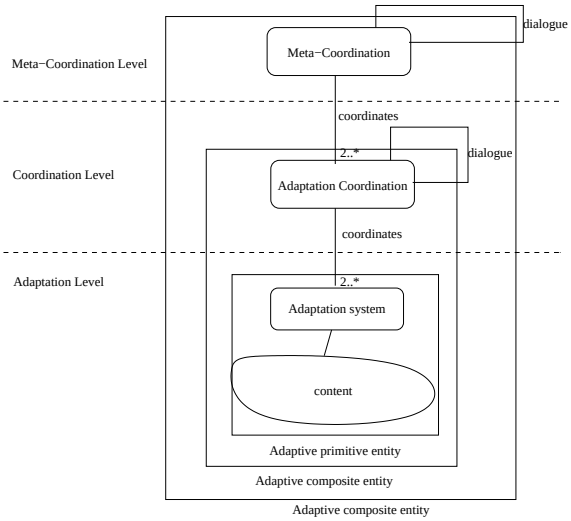

Figure 3: Distributed adaptation model.

ming model of the application is captured into the single implementation of the *execute* component, the DYNACO framework is highly generic. We therefore consider functionalities of DYNACO and their relations as our mandatory model. Each adaptive entity should include a specialization of DYNACO in order to implement adaptation.

4 DISTRIBUTED ADAPTATION

As mentioned in section 3.1, adaptation mechanisms may be associated to different entity levels leading to the primitive and composite adaptive entities. When associating adaptation mechanisms to the whole application, an adaptation designer must propose a holistic adaptation policy. This approach may be interesting for applications that do not need much adaptation and leads to a centralized adaptation architecture. However, for highly adaptive or distributed applications a better approach would be to partition the adaptation policy by considering adaptation for each entity separately and then coordinate adaptation performed on different entities.

Separation between adaptation and adaptation coordination may be applied at different levels depending on the complexity of the adaptation policy to be implemented. According to this, Figure 3 presents our model for distributed adaptation. Each adaptive primitive entity has an adaptation system associated to it that decides, plans, and executes adaptation for it. On the other hand, adaptive composite entities encapsulate an adaptation coordination system that coordinates adaptation of different adaptive primitive entities and dialogues with other systems so that global adaptation is performed. Alternatively, communication among adaptation coordination systems may be governed by a meta-coordination system. Although the tower for coordination may have infinite layers, a pragmatic approach may limit it to two or three.

4.1 Adaptation for Data Management

In order to validate our distributed adaptation model, we have considered a (highly) simplified version of our ambient assistive living application and in particular a data consistency management service (CMS). This service may be used by medical staff in the same or different hospitals, mobile or not, to maintain consistency among replicas of patients' records. Depending on the execution context[3], different consistency protocols may be used. For example, when high network bandwidth is available, a strong consistency protocol is well suited as it ensures that all replicas are consistent anytime and the user will not observe delays on reading or writing. On the other hand, when low network bandwidth is available, a weak consistency protocol is better suited as it allows replicas to be accessed even if network partitions exist.

For scalability and fault tolerance reasons, the CMS has been implemented as a distributed service. An update entity is associated to each replica that allows accessing (reading and writing) the local replica and notifying other replicas about local updates. The adaptation policy to be implemented advocates the use of the strong consistency protocol when possible (high bandwidth availability).

Traditionally, consistency management services provide a single protocol to manage different replicas. Indeed, they assume the environment[4] as a whole and propose the protocol that better suits the context offered by it, i.e. the "worst case". For example, even if several replicas are well-connected, the Bayou replicated database will use a weak consistency protocol for all replicas if one of them is weakly connected (Terry et al., 1998). In order to avoid such a problem, we consider the environment as a set of sub-environments each of which 1) offers a context, and 2) does not overlap other sub-environments, i.e. devices on two sub-environments do not share network connections with the same properties.

Figure 4 shows our CMS and the adaptation architecture associated to it. For simplicity reasons, the figure shows replicas (four) of one particular data. The environment is divided into two sub-environments.

[3] Available memory, CPU, bandwidth, etc.

[4] Set of devices and network connections that use the service.

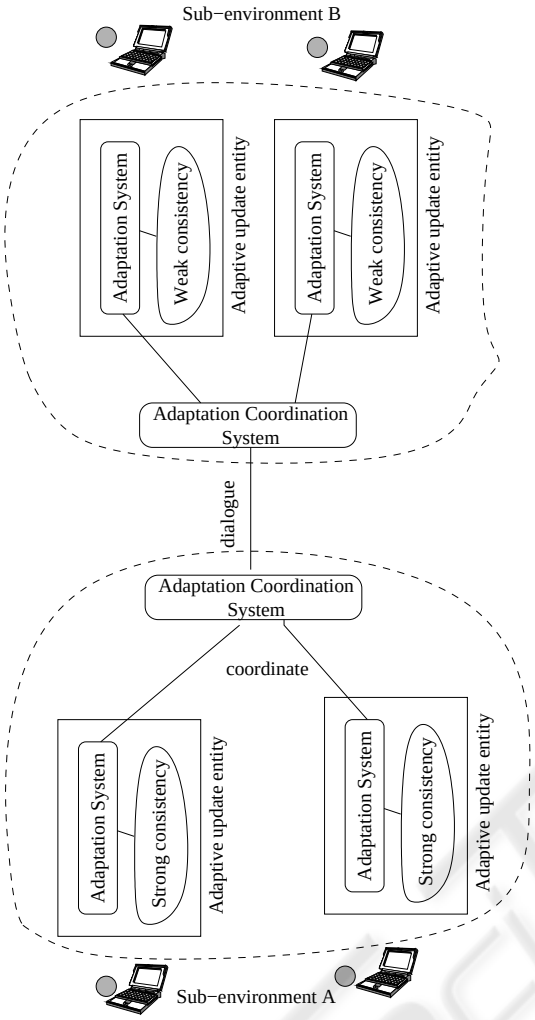

Figure 4: Distributed adaptation architecture for consistency management.

Sub-environment A offers high network bandwidth; replicas on it are managed using a strong consistency protocol. On the other hand, replicas on sub-environment B are managed by a weak consistency protocol as bandwidth availability is low.

As shown in the figure, the adaptation architecture is two-layered. Each update entity has an adaptation system associated to it that changes from strong to weak consistency and vice-versa on bandwidth availability changes. Coordination is performed at the sub-environment level; adaptation on each sub-environment is managed by an adaptation coordination system that dialogues with its counterpart on the other environment in order to perform global adaptation. The architecture respects our adaptation model. Each update entity is modeled as an adaptive primitive entity. Update entities on an environment are encapsulated in an adaptive composite entity and their adaptation is coordinated by the Adaptation Coordination System.

5 RELATED WORK

Several architectures and frameworks have been previously proposed to implement dynamic adaptation. These architectures usually emphasize the requirement for reflective programming support in order to implement the adaptation actions, such as in (Amano and Watanabe, 1999), (Keeney and Cahil, 2003), and (Capra et al., 2001). Indeed, reflexivity provides means to modify application behavior, in the underlying runtime environment or in the adaptation framework itself. In our approach, the need for reflexivity is captured within adaptation actions. In fact, our model does not specify how actions are implemented; nevertheless, parametrability, reflexivity and runtime code modification are the major tools. Moreover, neither of these works takes into account distribution of the adaptation mechanisms which limits their interest.

Other approaches such as (Lapayre and Renard, 2005) and (Vadhiyar and Dongarra, 2005) are based on the runtime environment and/or compilers to make applications adaptive. Therefore, the set of possible adaptation strategies is restricted by the developers of the runtime environment and/or compilers. In our approach, the adaptation developer is responsible for customizing the framework according to application needs.

In (Yan and Sere, 2004) formalizations of dynamic adaptation are proposed to verify correctness of adaptation. However, they do not provide help in designing the adaptation itself. This approach is complementary to ours, which focuses on helps for designing the adaptation while dismissing the problem of the verification.

Only few studies have been conducted about the methodology of dynamic adaptation. In (McIlhagga et al., 1998), authors list relevant questions that developers should answer to design adaptive software. However, this work does not provide a clear structure of an adaptive application, potentially leading to *ad-hoc* implementations and poor reusability. Moreover, distribution of adaptation is not taken into account.

6 CONCLUSIONS AND FUTURE WORK

In this paper we have presented a model for structuring dynamic adaptation mechanisms intended to be

used in a distributed environment and its utilization for a consistency service used in an ambient assistive living application. The model assumes applications as a set of "entities" (components, services) which may include adaptation mechanisms as a set of functionalities. Some of these functionalities are mandatory and relate to decide, plan and execute adaptation actions; others are optional and relate to coordinate adaptation actions on different entities. Indeed, depending on the complexity of the adaptation policy, designers are free to partition the policy so that several entities encapsulate their own one. Coordination may be necessary to ensure a consistent global adaptation.

We have already implemented and tested a centralized version of our model, without coordination mechanisms (Buisson et al., 2006). Ongoing work concerns the implementation of a first prototype of our model applied to the consistency service as described in section 4. The prototype is based on the Fractal component model and its Java implementation Julia (Jul,) as its hierarchical model fits well into our primitive/composite entities. The prototype will allow us to better understand how to propagate and coordinate changes through the system, evaluate costs and benefits of distribution, and to precise how to structure an environment (or an adaptation framework) in face of overall dynamic properties. Moreover, new functionalities may be identified and the coordination/meta-coordination should be refined in order to clearly separate decision, planning, and execution coordination.

REFERENCES

Jess, the rule engine for the java platform. herzberg.ca.sandia.gov/.

Julia home page. fractal.objectweb.org/julia/index.html.

Amano, N. and Watanabe, T. (1999). An approach for constructing dynamically adaptable component-based software systems using lead++. In *OOPSLA'99 International Workshop on Re ection and Software Engineering*.

Börsch-Supan, A. and Jürges, H. (2005). The survey of health, ageing and retirement in europe - methodology. Technical report, Mannheim Research Institute for the Economics of Aging.

Bruneton, E., Coupaye, T., Leclercq, M., Quéma, V., and Stéphani, J.-B. (2006). The fractal component model and its support in java. *Software Practice and Experience, , special issue on Experiences with Auto-adaptive and Reconfigurable Systems*.

Buisson, J. (2006). *Adaptation dynamique de programmes et composants parallèles*. PhD thesis, INSA de Rennes, IRISA.

Buisson, J., André, F., and Pazat, J. (2006). Afpac: Enforcing consistency during the adaptation of a parallel component. *Scalable Computing: Practice and Experience*.

Capra, L., Emmerich, W., and Mascolo, C. (2001). Reflective middleware solutions for context-aware applications. In *Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns*.

Commission, E. (2007). Ageing well in the information society - an i2010 initiative - action plan on information and communication technologies and ageing. Technical report, European Commission.

Fikes, R. and Nilsson, N. (1971). Strips: A new approach to the application of theorem proving. *Artificial Intelligency*, 2.

Keeney, J. and Cahil, V. (2003). Chisel: a policy-driven, context-aware, dynamic adaptation framework. In *4th International Workshop on Policies for Distributed Systems and Networks*.

Lapayre, J. and Renard, F. (2005). Appat: A new platform to perform global adaptation. In *International Conference on Distributed Frameworks for Multimedia Applications*.

McIlhagga, M., Light, A., and Wakeman, I. (1998). Towards a design methodology for adaptive applications. *Mobile Computing and Networking*.

Terry, D., Petersen, K., Spreitzer, M., and Theimer, M. (1998). The case for non-transparent replication: Examples from bayou. *IEEE Data Engineering*.

Vadhiyar, S. and Dongarra, J. (2005). Self adaptability in grid computing. *Concurrency and Computation: Practice and Experience*.

Yan, L. and Sere, K. (2004). Formalism for context-aware mobile computing. In *Third International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks*.